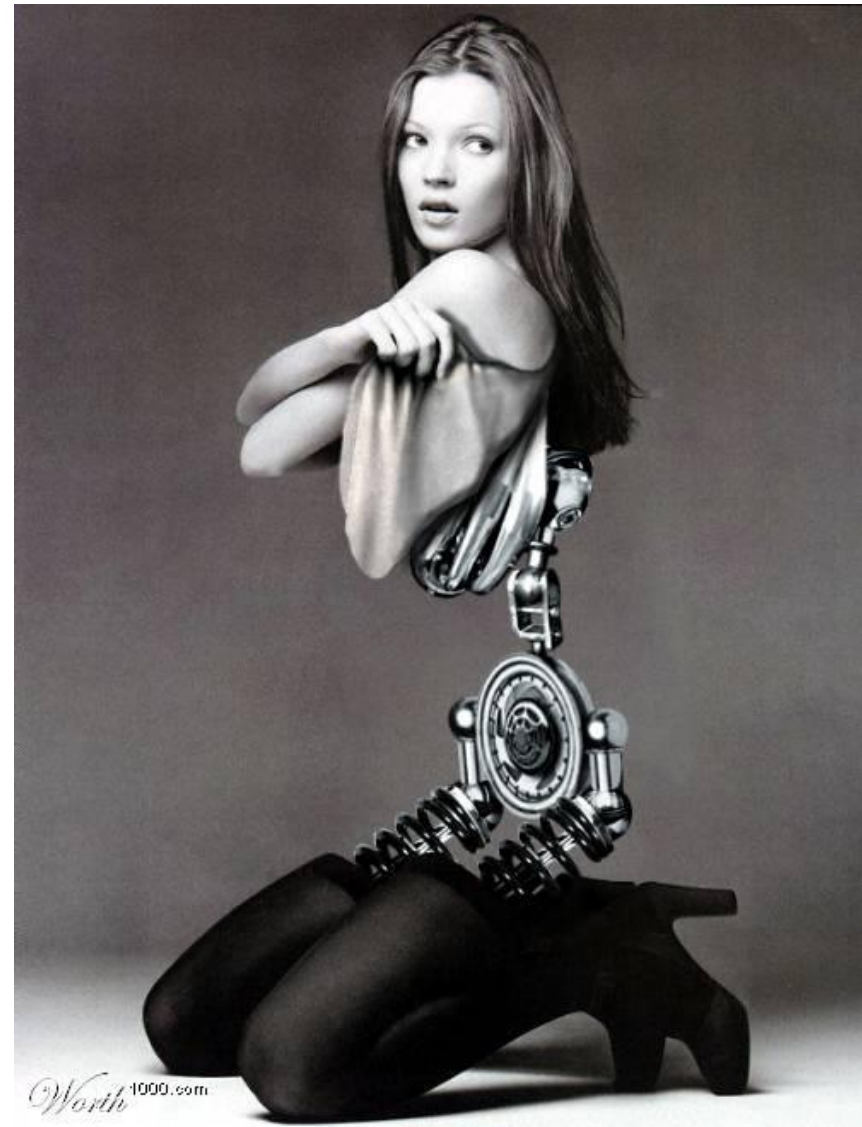


# Sissejuhatus Infotehnoloogiasse

## Tehismõistus



# Loengu ülevaade

---

- Eesmärgid
- Algusaegade naiivoptimism
- Miks on AI nii keeruline?
- AI filosoofia, Turingi test, pseudointellekt ja Eliza
- Kaks äärmust: kunstlikud putukad ja paljuteadev CYC
- Loogikapõhised meetodid, teadmised ja järeldamine
- Kõnest ja piltidest aru saamine
- Robootika
- Teadmiste integratsioon
- Mõtlemismängude otsialgoritmid

# Taust ja põhimõtted: eesmärgid

---

Erinevad eesmärgid, üldiselt konkreetsele:

- Tehisintellektinduse “**suur eesmärk**” ehk „strong AI“ on päriselt intelligentse masina ehitamine: riistvara + tarkvara
- Tehisintellektinduse “**filosoofiline eesmärk**” on saada paremini aru mõistuse (sh inimese ja loomade) funktsioneerimise põhimõtetest üldse.
- Tehisintellektinduse “**praktiline eesmärk**” ehk „weak AI“ on teha programme, mis oskaksid lahendada keerulisi ülesandeid, mille jaoks siiani on inimesi vaja.

# Strong AI:

“if a machine approaches or supersedes human intelligence, if it can do typically human tasks, if it can apply a wide range of background knowledge and has some degree of self-consciousness”

# Weak AI:

“the use of software to study or accomplish specific problem solving or reasoning tasks that do not encompass (or in some cases, are completely outside of) the full range of human cognitive abilities. “

## Ajaloo algus, naiivoptimism ja fiaskod

- The field of artificial intelligence research was founded at a conference on the campus of Dartmouth College in the summer of 1956.
- Those who attended would become the leaders of AI research for decades.
- Many of them predicted that a machine as intelligent as a human being **would exist in no more than a generation and they were given millions of dollars to make this vision come true.**

## Kuuekümnendate reaalsed arengud:

- Väga piiratud keelt mõistvad robotmanipulaatorid, mis suudavad veidi tegevust planeerida ja kaste laduda.
- Lihtsakoelisi loogikajäreltusi teha suutvad süsteemid.
- Esimesed male- ja kabeprogrammid.

## „AI freeze“:

- Eventually it became obvious that they had grossly underestimated the difficulty of the project. In 1973, in response to the criticism of Sir James Lighthill and ongoing pressure from congress, the U.S. and British Governments stopped funding undirected research into artificial intelligence.
- Seven years later, a visionary initiative by the Japanese Government inspired governments and industry to provide AI with billions of dollars, but by the late 80s the investors became disillusioned and withdrew funding again.



## Oluline aru saada:

- Inimene ei teadvusta oma ajutegevust.
- Inimesele paistab ekslikult, et paljud tema igapäevased toimingud on väga lihtsad ja ei nõua palju teadmisi ja ülikeerukaid ajuprotsesse

## Oluline aru saada:

- „Lihtne tehisintellekt“ ei ole võimalik.
- Süsteemi võimsamaks tegemiseks on paratamatult vaja lisada üha rohkem ja rohkem erinevate spetsiaalsustega komponente.

# Kõrvalefektid tulemustena

- Tehisintellekti-uuringud on andnud hulgaliselt algoritme ja meetodeid ja programmeerimiskeeli, mida rakendatakse praktikas mitte-tehisintellekti-ülesannete jaoks. Näiteks:
  - Paljud otsimisalgoritmid
  - Paljud optimeerimisalgoritmid
  - Formaalsete keelte süntaksianalüüs (kõigis kompilaatorites)
  - Funktsionaalsed ja loogilised programmeerimiskeeled
  - Objekt-orienteeritud programmeerimine
  - Lausearvutuse valemite ja analoogiliste ülesannete efektiivne lahendamine
  - ... jne
- Niipea, kui mingit seni väga rasket ülesannet osatakse programmiga efektiivselt lahendada, nihkub ülesanne tehisintellekti-uuringute vaateväljast minema.
- Tehisintellekti-uuringud tegelevad seepärast reeglina ülesannetega, mida veel eriti hästi programmide abil lahendada ei osata.

# Taust ja põhimõtted: arvutatavusest

---

- Meenutame, et paljude ülesannete jaoks ei saa olla lahendavat algoritmi.  
See kehtib nii:
  - arvuti jaoks
  - inimese jaoks
- Aga:
  - Inimene “saab lahendada” mistahes ülesande, kuna ta teeb vahel vigu.
  - Arvuti saab ka “lahendada” mistahes ülesande, kui programmis kasutada juhuslike arvude generaatorit ja panna programm vahel vigu tegema.

Kahte sorti:

- **Üldised küsimused (puhas filosofeerimine):**

- kas on võimalik teha mõistuslikku masinat?
- mis asi on mõistus?
- mis asi on teadvus?
- .. jne

- **Konkreetsed küsimused (rakenduslik filosofeerimine):**

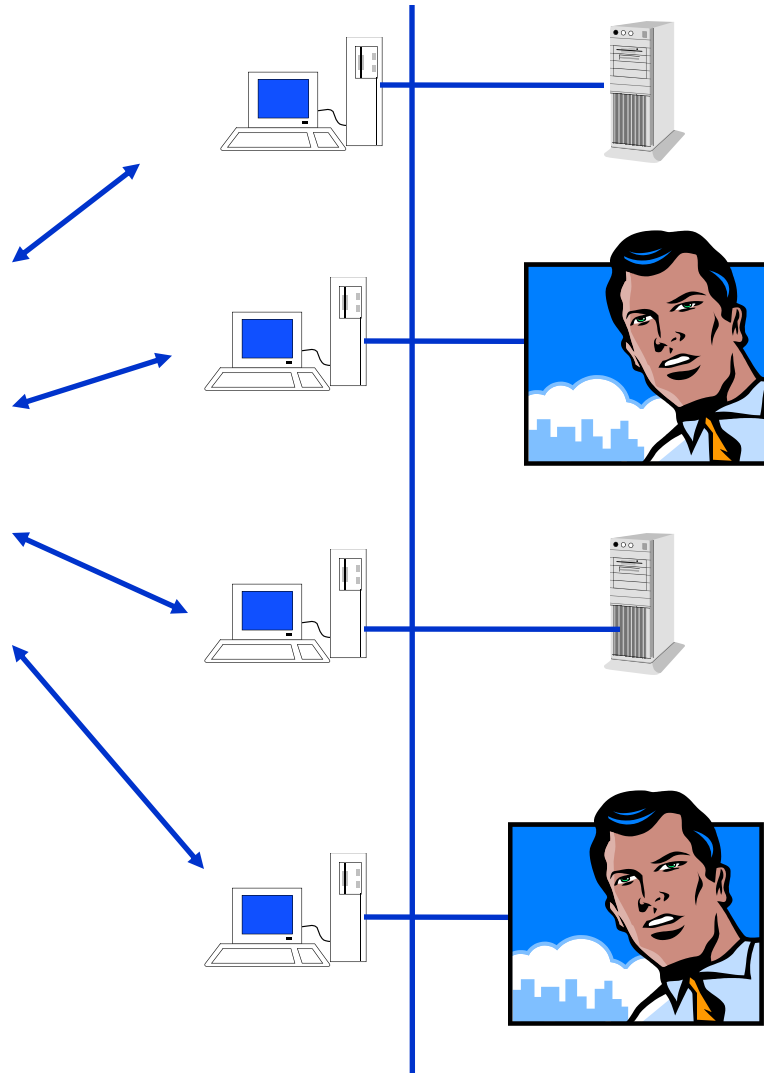
- kuidas inimene teise inimese öeldud lausest aru saab?
- mis on tähenduse mehhanismid?
- kuidas inimene kasutab oma teadmisi igapäevases elus hakkamasaamiseks?
- kas mõtlemismehhanismid on kollektiivsed või pigem ühe algoritmi poolt läbiviidavad?
- ... jne

# Filosoofia: üldküsimused ja Turingi test

- Turingi test: Turingi filosoofiline idee/ettepanek 30-datest:



Mõistata, kas  
chati-ekraani  
taga on inimene  
või programm?



Turing:  
Kui katsetajad  
ei suuda ära  
arvata (st  
ära-arvamise  
sagedus on  
50% ja 50%  
eksitakse), siis  
on jutlev  
masin  
**päriselt**  
intelligentne.

# Filosoofia ja praktika Turingi testi ümber

- Palju erinevaid vastuväiteid, et testi ärategemine ei tähenda veel “päris” intellekti.
- Ükski vastuväide ei ole ümberlükatav.
- Tegelikkuses ei suuda praegu keegi teha programmi, mis suudaks Turingi testi edukalt läbi teha: sellest ollakse VÄGA KAUGEL.
- Praegu “edukalt” intellekti simuleerivad Turingi testi programmid on kas:
  - hullumeelse simulaatorid:
    - katatoonik: ei ütle üldse mitte midagi
    - paranoik: räägib kogu aeg samasugust hirmujuttu
  - poolsegase psühhoanalüütiku simulaatorid (Eliza)
  - jms simulaatorid, mis hoiavad keskustelu VÄGA KITSASTES raamides ja sisuliselt ei saa üldse aru talle öeldud lausetest.
- Veidi põhjalikuma vestlusega saab alati aru, et simuleerival programmil pole üldist arusaama isegi igapäevasest elust ja asjadest ning ta ei oska probleeme lahendada.

# Vestlev pseudointellekt: Eliza

---

- In 1966 **Joseph Weizenbaum** at M.I.T. wrote **ELIZA**, a famous program that simulates a Rogerian psychoanalyst by taking excerpts from the subject's comments and posing questions back to the subject.
- Eliza on lihtne programm, mille sees on suur hulk “standardlauseid” ja “standardsõnu”. Kui Elizale anda lause, milles on sõna tema “standardsõnade” loetelus, siis Eliza valib mõne “standardvastuse” ja paneb sinna sisse osi talle antud tekstist.
- Jääb mulje, et Eliza haakub vestlejaga. Eliza poolt antud laused on samuti harilikud inimkeelsed laused (sest need on programmeerija talle andmebaasi pannud”).



# Eliza: seletustega näitevestlus

ELIZA: HELLO, I'M ELIZABETH. WHAT WOULD YOU LIKE TO TALK ABOUT?

*(the Welcome message)*

**USER: My mum is having a hard time.**

ELIZA: TELL ME MORE ABOUT YOUR FAMILY

*(line 5 changes 'mum' into 'mother', and this is matched at line 26; hence one of lines 28, 29, and 30 is randomly selected)*

**USER: There's just her, dad and my brother**

ELIZA: WHAT DO YOU REMEMBER MOST ABOUT YOUR CHILDHOOD?

*(line 6 changes 'dad' into 'father', and this is matched at line 27; hence either line 29 or 30 is randomly selected for display – line 28 is ruled out because it was the last response to be selected from this group)*

**USER: My father and brother were always quarrelling.**

ELIZA: ARE YOU THE YOUNGEST IN YOUR FAMILY?

*(‘father’ is matched at line 27; hence either line 28 or 29 is randomly selected for display – line 30 is ruled out because it was the last response to be selected from this group)*

**USER: My brother is younger than me.**

ELIZA: SO YOU ARE OLDER THAN YOUR BROTHER.

*(line 31 matches, so line 32 then generates the response 'SO me IS OLDER THAN my brother'. Output substitutions then transform this in turn to 'SO YOU IS OLDER THAN my brother' (line 10), 'SO YOU IS OLDER THAN YOUR brother' (line 13), and finally 'SO YOU ARE OLDER THAN YOUR brother' (line 18).*

# Filosoofia ja praktika Turingi testi ümber

---

- Tegelikult Turingi testi edukalt tegev programm peaks:
  - Teadma igapäevaelust sama palju, kui keskmine inimene
  - Oskama lahendada ülesandeid
  - Saama aru vestluspartneri soovidest ja teadmistest jne
- See kõik on VÄGA KEERULINE ja ei ole põhjust arvata, et lihtsakoeline programm sellega kuidagi hakkama saab.
- Kui programm sellega hakkama saab, siis on ta pööraselt keeruline ja võimas, ja inimesed üldiselt ei saa enam aru, mis ta sees on ja kuidas ta töötab, ja pigem võib arvata, et ta on siis ka päriselt intelligentne.

# Argumente masinmõistuse võimalikkuseks

- Arvutiprotsessorit mikroskoobiga vaadates on näha ränist juhtmerägastikku, mitte mõistust. Aju mikroskoobiga vaadates on näha valkudest neuronite rägastikku, mitte mõistust.
- Kas inimene suudab teha midagi endast palju võimsamat? Aga, kas ainurakne suudab teha hulkrakset? Kõrgemat looma?
- Kui mõistuslik programm on teoreetiliselt võimalik, siis võib ta luua ka juhuse läbi, saamata täpselt aru, miks ja kuidas.
- Religioosne problemaatika: kanoonilised tekstid ei ütle, et mingit uut mõistuslikku eluvormi ei saa luua. Kui oletada, et seda saab teha ainult kõrgem olend, siis võib tal olla seda mugav teha inimeste kaudu.
- Ei ole mingit põhjust arvata, et inimene poleks masin. Kui oletada, et inimene kasutab ekstrasensiitvseid võimeid ja teda juhib kõrgem olend, siis miks ei võiks ka ränist masin kasutada ekstrasensitiivseid võimeid ja miks ei võiks teda samuti juhtida kõrgem olend?

# Teadvuse küsimused

---

- Teadvus on väga pinnaline. Inimene ei ole teadlik rõhuvast enamusest oma ajutegevusest, mõtlemise printsiipidest jne jne. Isegi maletaja ei oska adekvaatselt selgitada, miks on üks käik parem ja kuidas ta käike valib.
- Teadvus on tihedalt seotud mälu ehitusega.
- Loomadel ja vastsündinutel ei ole teadvust inimese mõttes.
- On olemas lõhestunud isikuga inimesi, kellel on mitu sõltumatut teadvust.
- Mõistust võib ehk vaadata kui riiki, kus sees palju mehhanisme ja tegutsejaid, ning mille presidendiks ehk esindajaks on teadvus, mida võib vajadusel vahetada (a la lõhestunud isikuga inimesed).

# Äärmused: putukad ja CYC

## ■ Putukad

- Rodney Brooks'i projekt MIT-s:  
teha putuka analoogideks olevaid roboteid.



- Idee: kui oskame teha putuka intellektiga masinat, siis proovime sealt edasi liikuda väikese looma intellektiga masinani, sealt kõrgema loomani, sealt inimeseni ja edasi.
- Lootus: sel viisil ehitame mõistuse mehhanismid analoogiliselt nende tekkele looduses.
- <http://www.ai.mit.edu/people/brooks/projects.shtml>

## ■ CYC

- Doug Lenati projekt CYC: teeme hiiglasliku andmebaasi faktidest ja reeglitest ja lihtsa järelduste tegemise programmi sinna peale.
- Idee: programm hakkab tekstidest aru saama, uusi reegleid ja fakte õppima, seejärel ka iseennast täiustama.
- Lootus: kvantiteet tekitab teatud piiri ületamisel kvalitatiivse hüppe.
- <http://www.cyc.com/>

# Järelduste tegemine

---

## ■ Peamine põhimõte:

- Loogikakeel on universaalne: selles saab esitada matemaatikat, igapäevaelu jms.
- Salvestame faktid ja reeglid sobivas loogikakeeles.
- Ehitame järelduste tegemise programmi, mis suudaks kiirelt kontrollida, kas meid huvitav järeldus antud faktidest ja reeglitest tegelikult tuleneb.

## ■ Eesmärk:

- Automatiseerida keeruliste järelduste tegemist nõudvaid ülesandeid.

## ■ Tulemusi ja praktika:

- Järelduste tegemise programmid on abiks spetsiifilistes valdkondades, kus etteantud reeglite arv ei ole väga suur (teatud matemaatika osad, elektroonikaseadmete ja raudteesignalisatsiooni õigsuse kontroll jne).
- Järelduste tegemise programmid jäävad tüüpiliselt ajahäta, kui reegleid on väga palju: raske on otsustada, mis reegleid mis järjekorras läbi proovida.

## ■ Valdkonna nimetus: automated reasoning, automated theorem proving

- Tihedalt seotud valdkond: ekspertsüsteemid

# Lahendamine loogikatõestuse abil

---

- Ülesanne tuleb loogika keeles formuleerida.
- Küsimus tuleb ka loogika keeles formuleerida.
- Tõestaja asub tõestust otsima.
- Kui anda lõpmatult palju aega ja mälu, siis tõestaja lõpuks ka tõestuse leiab (kui tõestus üldse teoreetiliselt eksisteerib). Keeruliste ülesannete puhul võtaks see lootusetult kaua aega (meenuta keerukusklasse!)
- Kui tõestust ei ole, siis tõestaja enamasti jääbki seda otsima, teadma, et sellist tõestust ei saa olla.

# Masinaga matemaatikaprobleem

W. McCune 1996:

- The Robbins problem---are all Robbins algebras Boolean?- has been solved using his automated theorem prover EQP.
- Programm otsis lahendust ca üks nädal, kuni lõpuks leidis.
- Ülesanne oli matemaatikute poolt lahendamata, kuigi püstitati aastal 1933:
- Meil on antud Robbinsi algebra:

$$x + y = y + x. \quad [\text{commutativity}]$$

$$(x + y) + z = x + (y + z). \quad [\text{associativity}]$$

$$n(n(x + y) + n(x + n(y))) = x. \quad [\text{Robbins equation}]$$

- Kas järgmised võrdused annava selle algebra jaoks vastuolu:

$$x+y \neq x.$$

$$n(x+y) \neq n(x).$$



# Tõestuse algus

2 []  $x+y=y+x$ .

3 []  $(x+y)+z=x+y+z$ .

4 []  $(x+y)+z=x+y+z$ .

5 []  $n(n(n(x)+y)+n(x+y))=y$ .

6 []  $n(x+y) \neq n(x)$ .

64 [para\_into,4.1.1.1,2.1.1,demod,3]  $x+y+z=y+x+z$ .

71 [para\_into,5.1.1.1.2.1,2.1.1]  $n(n(n(x)+y)+n(y+x))=y$ .

73 [para\_into,5.1.1.1,2.1.1]  $n(n(x+y)+n(n(x)+y))=y$ .

75 [para\_into,6.1.1.1,2.1.1]  $n(x+y) \neq n(y)$ .

93 [para\_into,71.1.1.1,2.1.1]  $n(n(x+y)+n(n(y)+x))=x$ .

116 [para\_into,75.1.1.1,4.1.1]  $n(x+y+z) \neq n(z)$ .

130 [para\_into,93.1.1.1.2,73.1.1]  $n(n(n(n(x)+y)+x+y)+y)=n(n(x)+y)$ .

132 [para\_into,93.1.1.1.2,5.1.1]  $n(n(n(x+y)+n(x)+y)+y)=n(x+y)$ .

139 [para\_into,116.1.1.1.2,64.1.1]  $n(x+y+z+u) \neq n(y+u)$ .

170 [para\_from,130.1.1,73.1.1.1.2,demod,3,3]  $n(n(n(n(x)+y)+x+y+y)+n(n(x)+y))=y$ .

174 [para\_from,130.1.1,5.1.1.1.1,demod,3,3]  $n(n(n(x)+y)+n(n(n(x)+y)+x+y+y))=y$ .

190 [para\_into,132.1.1.1.1.2,64.1.1]  $n(n(n(x+y+z)+y+n(x)+z)+y+z)=n(x+y+z)$ .

211 [para\_into,139.1.1.1,64.1.1]  $n(x+y+z+u) \neq n(x+u)$ .

253 [para\_from,170.1.1,93.1.1.1.2]  $n(n(n(n(x)+y)+n(n(x)+y)+x+y+y)+y)=n(n(x)+y)$ .

260 [para\_from,170.1.1,73.1.1.1.2.1.1,demod,3]  $n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+z)+n(y+z))=z$ .

346 [para\_into,190.1.1.1.2,2.1.1]  $n(n(n(x+y+z)+y+n(x)+z)+z+y)=n(x+y+z)$ .

....

....

423 [para\_from,253.1.1,5.1.1.1.1,demod,3,3,3,3]  $n(n(n(x)+y)+n(n(n(x)+y)+n(n(x)+y)+x+y+y+y))=y$ .

475 [para\_from,260.1.1,73.1.1.1.2,demod,3,3]  $n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+z+n(y+z))+z)=n(y+z)$ .

486 [para\_into,346.1.1.1.1.1.1.1,2.1.1,demod,3]  $n(n(n(x+y+z)+x+n(z)+y)+y+x)=n(z+x+y)$ .

684 [para\_into,475.1.1.1.1.1.2.2,2.1.1]  $n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+n(y+z)+z)+z)=n(y+z)$ .

801 [para\_from,684.1.1,73.1.1.1.2.1.1,demod,3]

$n(n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+n(y+z)+z)+z+u)+n(n(y+z)+u))=u$ .

848 [para\_into,801.1.1.1.1,486.1.1,demod,3]  $n(n(x+x+n(n(x)+x)+x)+n(n(x+x)+n(n(x)+x)))=n(n(x)+x)$ .

878 [para\_into,848.1.1.1.2,93.1.1]  $n(n(x+x+n(n(x)+x)+x)+x)=n(n(x)+x)$ .

897 [para\_into,878.1.1.1.1.1.2,64.1.1]  $n(n(x+n(n(x)+x)+x+x)+x)=n(n(x)+x)$ .

948 [para\_into,897.1.1.1.1.1,64.1.1]  $n(n(n(n(x)+x)+x+x+x)+x)=n(n(x)+x)$ .

1015 [para\_from,948.1.1,73.1.1.1.2.1.1,demod,3]  $n(n(n(n(n(x)+x)+x+x+x)+x+y)+n(n(n(x)+x)+y))=y$ .

1036 [para\_from,948.1.1,73.1.1.1.2,demod,3,3,3]  $n(n(n(n(x)+x)+x+x+x+x)+n(n(x)+x))=x$ .

1053 [para\_into,1015.1.1.1.2,423.1.1]

$n(n(n(n(n(x)+x)+x+x+x)+x+n(n(n(x)+x)+n(n(x)+x)+x+x+x+x))+x)=n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)$ .

1079 [para\_from,1036.1.1,93.1.1.1.2]  $n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x)=n(n(x)+x)$ .

1112 [para\_into,1053.1.1.1.1.1,2.1.1,demod,3]

$n(n(x+n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+n(n(n(x)+x)+x+x+x))+x)=n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)$ .

1130 [para\_from,1079.1.1,73.1.1.1.2.1.1,demod,3]  $n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x+y)+n(n(n(x)+x)+y))=y$ .

1149 [para\_into,1112.1.1.1.1.1,64.1.1]

$n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x+n(n(n(x)+x)+x+x+x))+x)=n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)$ .

1169 [para\_into,1130.1.1.1.2,174.1.1]

$n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x+n(n(n(x)+x)+x+x+x))+x)=n(n(n(x)+x)+x+x+x)$ .

1211 [para\_into,1169.1.1,1149.1.1]  $n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)=n(n(n(x)+x)+x+x+x)$ .

1212 [binary,1211.1,211.1] \$F.

- Ekspertsüsteem on mõne spetsiifilise valdkonna jaoks kohandatud järelduste tegemise programm (variant teoreemitõestajast)
- Tihtipeale kasutavad ebaharilikku, spetsiaalselt valdkonnaga sobitatud loogikat.
- Enamasti sisaldavad paljusid valmiskujul reegleid.
- Enamasti sisaldavad mugavat kasutajaliidest: kasutaja ei pea loogika keelt oskama.
- Tüüpiliselt kasutatakse mõne suure rakenduse osana

- Intelligentne agent on nagu robot ilma füüsiseta: ta on ainult programm, mis suhtleb välismaailmaga standardse programmi kombel:
  - Display
  - Klaviatuur
  - Hiir
  - HTTP ja muud võrguühendused
- Intelligentne agent peaks ise koguma võrgust infot, talletama kasutaja antud reegleid ja fakte, oskama vastata tema päringutele, tegema ise võrgust otsinguid jne.
- Kuulsad näited: Wolfram Alpha, Watson, SIRI

# Pildi tuvastamine

---

- Tehtud suurt progressi sõjatööstuse vajaduste tõttu.
- Tüüpiline ülesanne: tunda ära kaamerapildist vaenlase tankid:
  - Eristada neid veoautodest
  - Eristada neid oma tankidest
- Kasutatakse palju närvivõrkude abil õppimist.
- **Paneme tähele, et:**
  - Inimese poolt kogetud pilt on peamiselt tema aju poolt loodud, mitte aga “toores pilt”, mis tuleb silmanärvidest: viimane on väga halva kvaliteediga.
  - Inimese aju on võimeline täiesti iseseisvalt looma visuaalset pilti:
    - Unes
    - Hallutsinatsioonide käigus

# Kõne tuvastamine

---

- Kõne moondamine tekstiks on üllatavalt raske.
- Miks?
- Jällegi, kõrvanärvidest tulev info on ebakvaliteetne (nii ka mikrofonist).
- Raske on teha vahet kõnel ja taustamüral.
- Raske on “viia kokku” eri hääldustega eri inimeste poolt öeldud sõnu
- Kõne adekvaatseks tuvastamiseks on vaja temast **samal ajal aru saada!**
- Näiteks: tuvastage võõras keeles kõnet taustamürast, eraldage sealt sõnu!
- Ilmselt parim süsteem praegu: Apple SIRI

# Teksti mõistmine ja tõlkimine

---

- Teksti mõistmine tähendaks selle teisendamist “sisemisteks”, näiteks loogikakeelseteks faktideks ja reegliteks.
- Teksti tähendus sõltub tohutult kontekstist.
- Teksti tähenduse teisendamine sisemisteks reegliteks eeldab väga suurte teadmiste olemasolu maailmast ja teksti kirjutajast!
- Sama probleem on tõlkimise juures: kvaliteetne tõlge eeldab teksti mõistmist.
- Mittekvaliteetne tõlge on samas lihtne.
- Tuntuim tõlkija - Google translate – ei saa tekstist mitte midagi aru

- Sisuliselt süntees kogu tehisintellektindusest.
- Lisandub:
  - Mehaanika
  - Mehaanika kiire ja täpne juhtimine (füüsikalised arvutused)
- Seepärast on “vingeid” roboteid praegusaja tehnoloogiaga pea võimatu teha.
- Näiteks ei suuda keegi teha robot-tennisisti, kes natukenegi mängiks.
- Tehtud on (suhteliselt kehvasid) robot-lauatennisemängijaid.
- TTÜ projekt: [roboswarm.eu](http://roboswarm.eu)
- Tipptasemel robotid: Boston dynamics „big dog“ ja Darpa grand challenge autod.



## Darpa grand challenge 2004, 2005

212 km väikestel maanteedel, võitja aeg 6.54

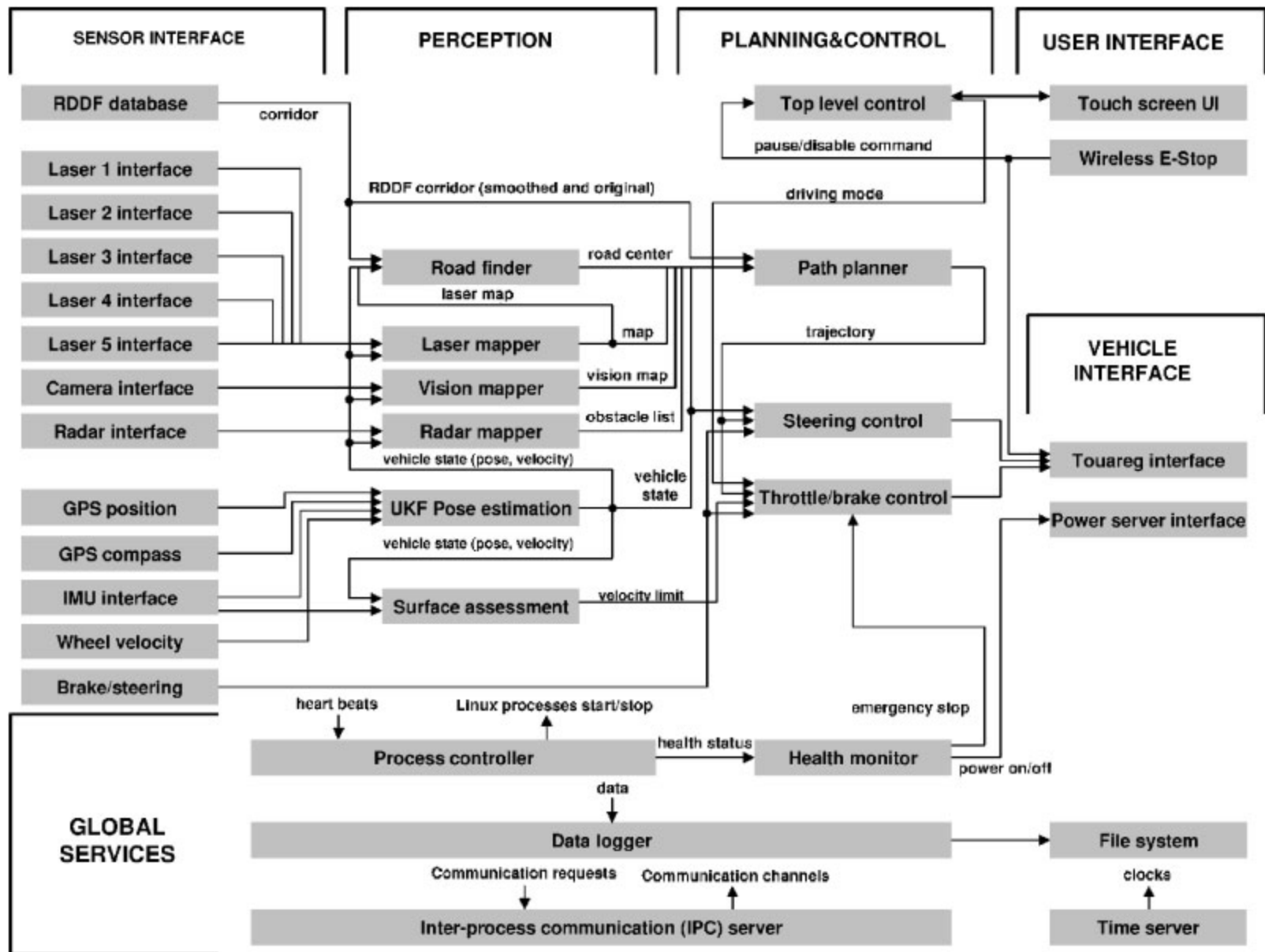


Stanley, Stanford racing team

## Darpa urban challenge 2007

96 km linnas, 4:10





At every time step, the planner considers trajectories drawn from a 2D space of maneuvers. The lookahead distance is speed dependent and ranges from 15 to 25 m.

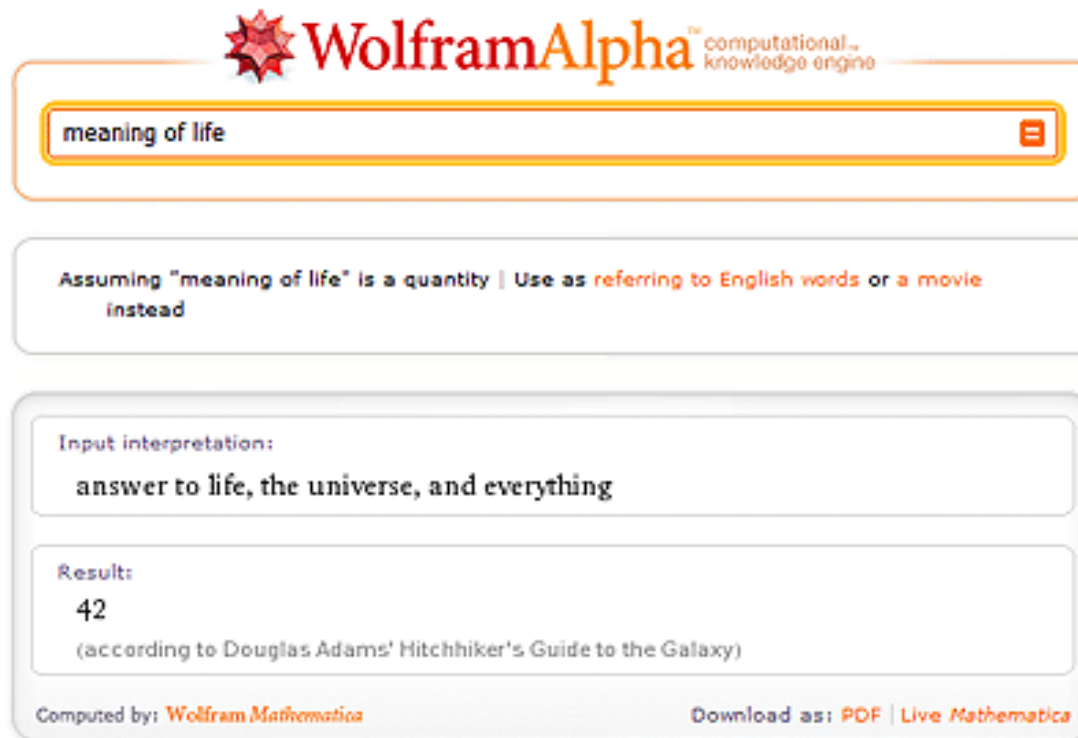
All candidate paths are run through the vehicle model to ensure that they obey the kinematic and dynamic vehicle constraints.


# Teadmiste integratsioon

---

- Olukord: võrgus tohutult palju inimloetavat infot, mis on masinaga lihtsalt kättesaadav
- Probleemid:
  - kuidas seda infot formaliseerida?
  - kuidas eri infojuppe seostada?
  - kuidas valida küsijale/kontekstile sobiv vastus?
- Lihtne näide: sightsplanneri probleem turismiobjektide samasuse tuvastamisel
- Tipptasemel süsteemid: Wolfram Alpha, Watson, SIRI

## Wolfram alpha online: 2009



 **WolframAlpha**<sup>™</sup> computational knowledge engine

meaning of life

Assuming "meaning of life" is a quantity | Use as referring to English words or a movie instead

Input interpretation:  
answer to life, the universe, and everything

Results:  
**42**  
(according to Douglas Adams' Hitchhiker's Guide to the Galaxy)

Computed by: [Wolfram Mathematica](#) Download as: [PDF](#) | [Live Mathematica](#)

## IBM Watson ja Jeopardy: 2011



## Mis on ülesanne?

- Meil on lauamäng (male, kabe vms), kus on kaks mängijat.
- Kumbki tahab võita.
- Teeme arvutiprogrammi, mis võtab seisu ette ja püüab leida võimalikult hea käigu sellest seisust.



# Veidi ajalugu

---

- Teise maailmasõja paiku:
  - Alan Turing tegi valmis maleprogrammi, aga tal ei olnud arvutit selle täitmiseks. Turing täitis programmi ise: paberi ja pliiatsi abil.
  - Mängis paar partiid: iga partii võttis hirmus kaua aega.
- 1940-1990:
  - Programmid läksid aeglaselt paremaks, arvutid kiiremaks, aga tipp-inim-mängijad olid programmidest kõvasti paremad.
- 1990-2000:
  - Programmid hakkasid tipp-inim-mängijaid järjepanu võitma
  - **CHINOOK** – 1994, 8x8 inglise kabe maailmameister
  - **Deep Blue** – 1997, võitis Garry Kasparovit malematshis
  - **LOGISTELLO** – 1997, võitis Othello (Reversi) maailmameistrit Takeshi Murakamit
- Praegune seis:
  - Mitmete mängude (Go, Bridzh, ...) jaoks on inim-mängijad endiselt programmidest hulga paremad

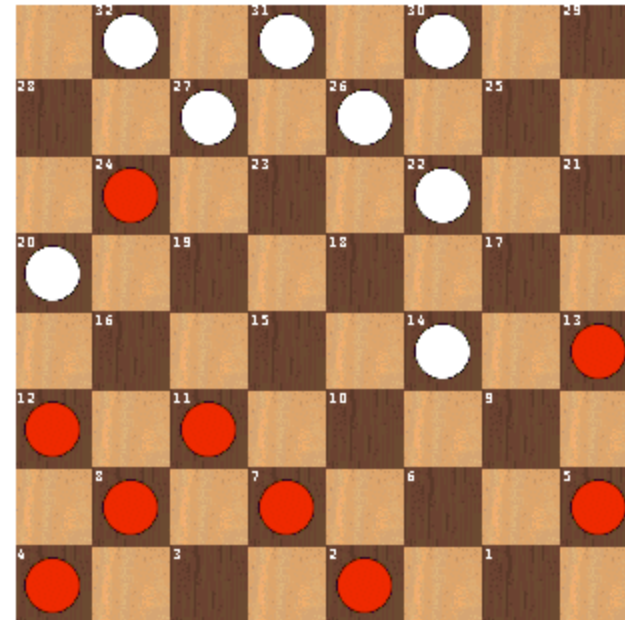
## Deep Blue versus Garry Kasparov: 1997, 3,5 x 2,5



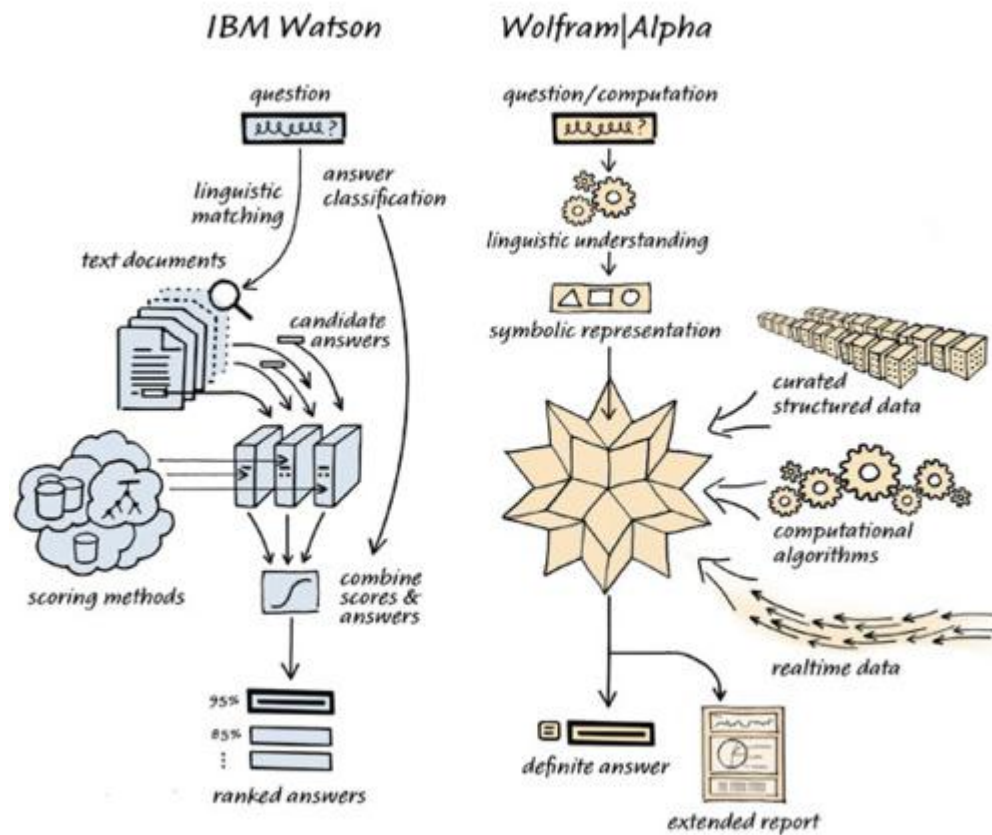
## Inglise kabe lahendatud: 2007, Chinook UAlberta

Checkers has a search space of  $5 \times 10^{20}$

Almost continuously since 1989, dozens of computers had been working around the clock to solve the game.



# Watson ja Wolfram alpha



# Kuidas maleprogrammi teha?

---

- Teeme kõigepealt programmi, mis:
  - loeb seisu
  - teeb seisu järgi mällu tabeli kõigist võimalikest käikudest selles seisus.
  
- See ei olegi nii väga keeruline programm
  - Males keerulisem
  - Kabes lihtsam
  - Othello (reversi) veel lihtsam
  - Viis nuppu ritta: väga lihtne
  
- Nüüd võib programm tabelist juhusliku käigu valida.
  
- Ja juba mängibki! Kuigi kehvasti ....

# Kuidas programmi veidi paremaks teha?

---

- Teeme eraldi väikese programmi, mis:
  - Võtab seisu ette
  - Ütleb umbes, kui hea seis on.
  - Seisu “headus” on number, mille meie programm arvutab.
- Kuidas öelda “kui hea”?
  - Vahel lihtne: kaotus / võit/ ei kumbki.
  - Numbrid:            1            -1            0
  - Keerulisem: “harilik maleseis”. Kumbalgi pole võitu ega kaotust.

- Keerulisem olukord: harilik maleseis.
- Materjal:
  - Loeme kokku meie ja vastase nupud.
  - Lipp annab 10 punkti, vanker 6, jne.
- Positsioon:
  - Loeme kokku eelviimasel real etturid masinal/inimesel
  - Loeme kokku tsentriväljadel olevad etturid masinal/inimesel
  - Loeme kokku, mitmele tsentriväljale masin/inimene tuld annab
  - Vaatame üle kuninga kaitse tugevuse
  - .... jne .....
- Kokku saame seisu headuse numbri:
  - Masina materjal – inimese materjal + masina positsioon – inimese positsioon

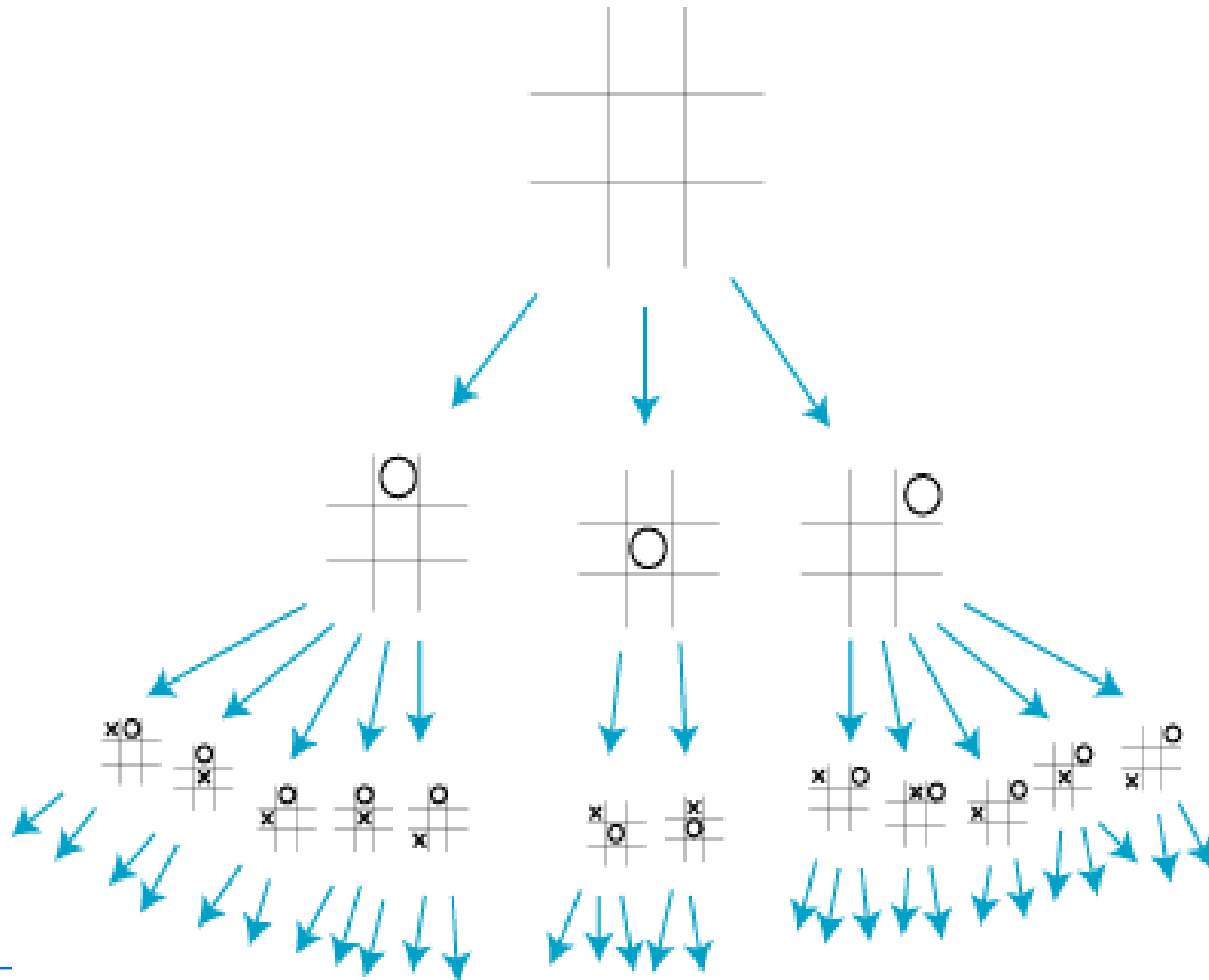
# Mis edasi?

---

- Nüüd on meil kõigi käikude tabel, iga käigu juures arvutame tekkiva seisu headuse numbri.
- Valime kõige parema tulemuse (maksimumi numbritest!) andva käigu.
- Programm hakkab veidi paremini mängima, aga:
  - Ikka mängib väga kehvasti!
  - Miks? Sest meie seisu headuse hindamise programm on nigel.
  - Kuidas headuse hindamist parandada?
- Loomulikult tuleks vaadata, mis käike vastane võib peale meie käike teha! Ja kuidas meie võime vastata. Jne.



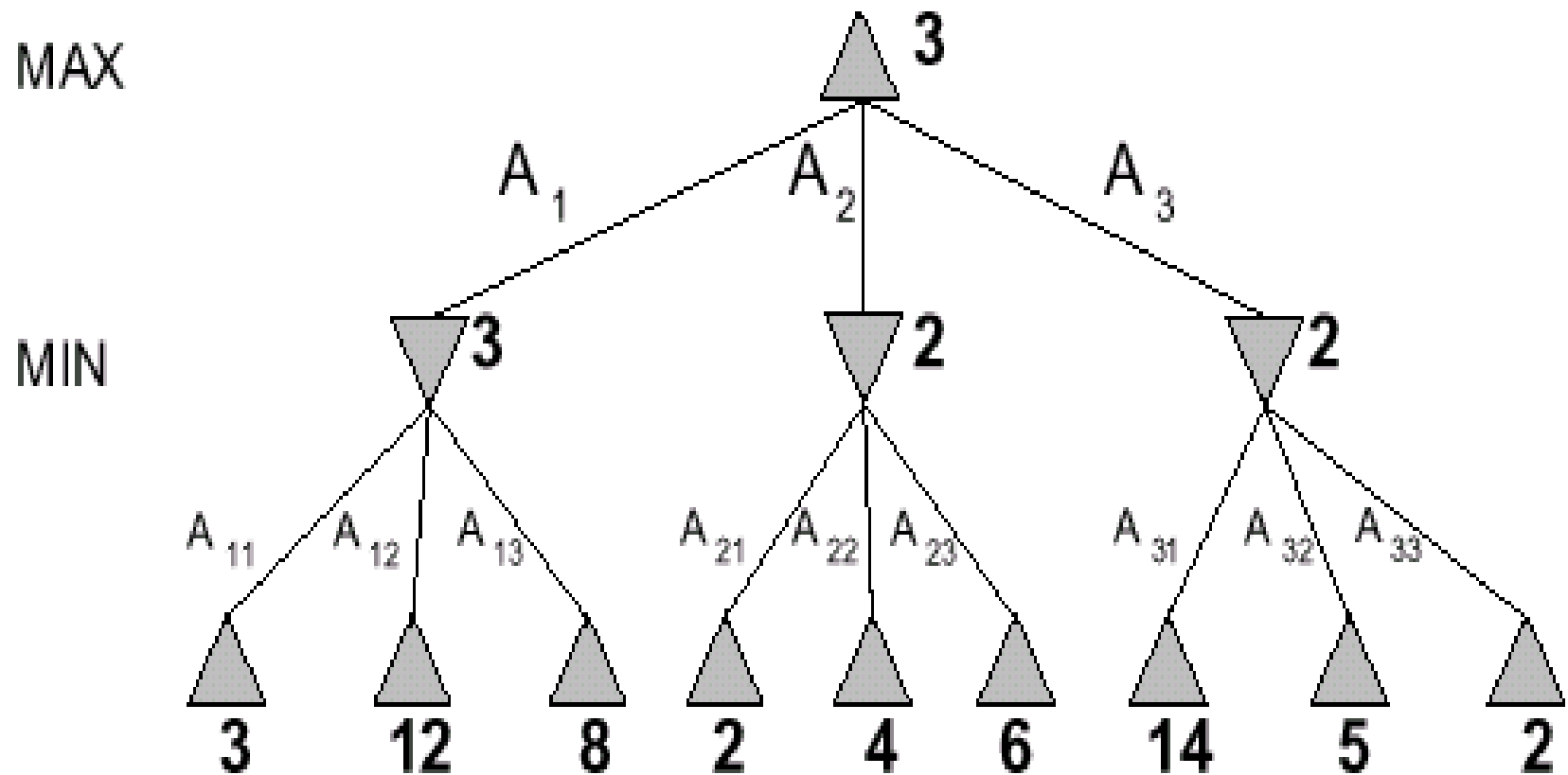
# käikude puu trips-traps-trulli näitel



# Kuidas sellist seisude puud kasutada?

- Eeldame lihtsalt, et:
  - masin tahab teha käiku, mis annab kõige suurema headuse numbriga seisu.
  - vastane tahab teha käiku, mis annab kõige väiksema headuse numbriga seisu.
- Seega:
  - igast seisust valib masin käigu, mis on maksimum-headusega (masinale).
  - igast seisust valib vastane käigu, mis on miinimum-headusega (masinale).
- Idee:
  - Vaatame käikude puud sügavuseni  $N$  (näiteks  $N=3$ )
  - Kõige alumistel seisudel arvutame lihtsalt headuse välja
  - Seejärel “tõstame” headuse numbreid ülespoole!

# Minimax algorithm

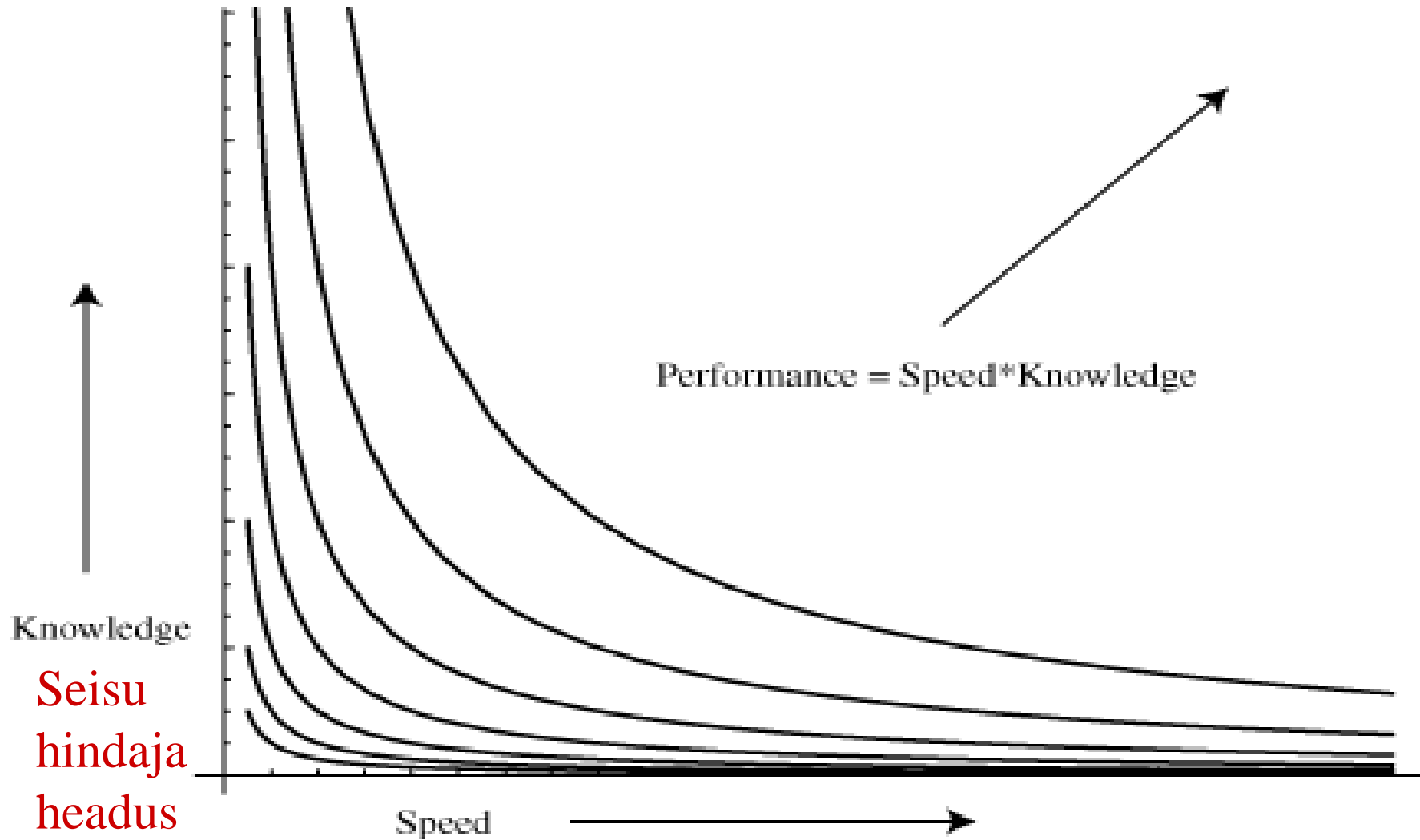


# Kui heaks programmi saab?

---

- Mida sügavamalt puud masin läbi jõuab vaadata, seda täpsemini ta käiku oskab valida.
- Puu läheb kiiresti väga suureks!
- Males ca 30 käiku ühes seisus.
  - **Esimesel tasemel käike 30.**
  - **Teisel tasemel käike  $30 \times 30$**
  - **Kolmandal tasemel käike  $30 \times 30 \times 30$**
  - ....
  - **N-ndal tasemel käike 30 astmes N.**
- Viiekümnendal tasemel oleks käike ca 30 astmes 50. See on rohkem, kui elementaarosakesi universumis!

# Umbes nii: teadmised ja kiirus ja mängu headus



Kui suure puu jõuab läbi vaadata

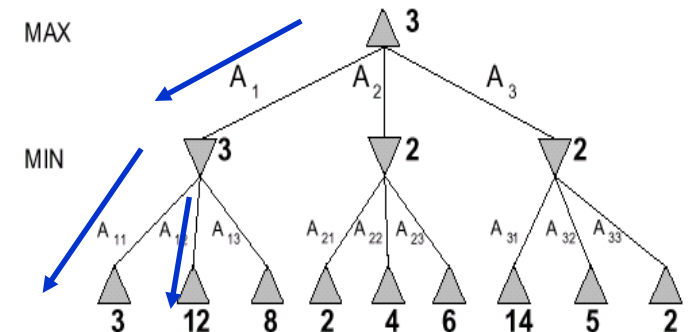
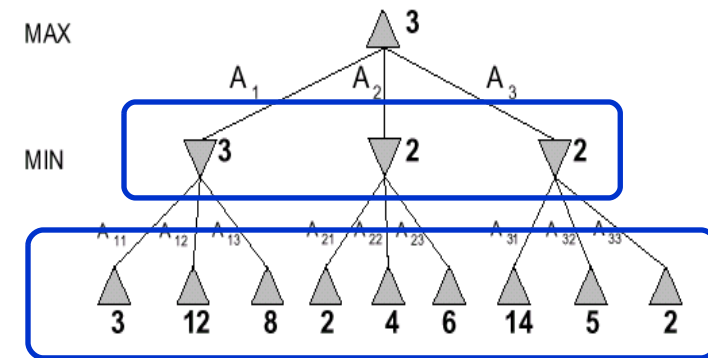
# Kuidas programmi parandada?

---

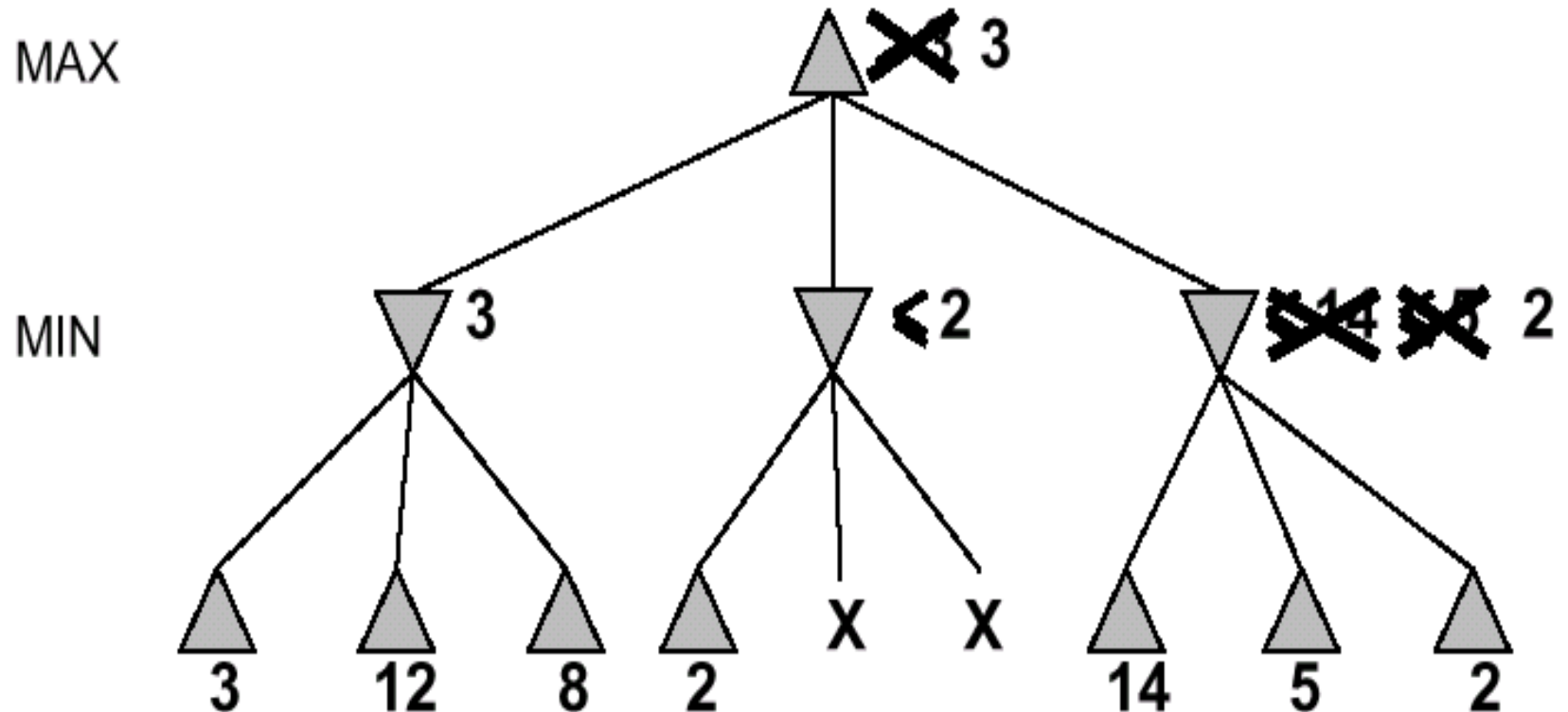
- Ei ole ühte head lahendust. On palju erinevaid nõkse!
- Näiteks:
  - Teeme seisu hindaja paremaks (programmi “targemaks”).
  - Aga siis läheb ta aeglasemaks ka.
  - Seega jõuame vähem käike läbi vaadata.
- Enamik nõkse on seotud käikude puu vähendamisega: ei ole vaja kogu puud läbi vaadata.

# Laiuti vs sügavuti otsing

- Kaks võimalust:
  - Otsime puu läbi kiht-kihilt
  - Otsime puu läbi sügavuti, minnes alul vasakul maksimaalse sügavuseni
- Eelistatakse sügavuti otsingut!
  - Mälu vaja palju vähem
  - Muud eelised ka



# Üks esimesi universaalseid meetodeid: alpha-beta





# Koodinäiteid: rekursiivsed minimax ja alpha-beta

```
int NegaMax (pos, depth)
{
    if (depth == 0) return Evaluate(pos);
    best = -INFINITY;
    succ = Successors(pos);
    while (not Empty(succ))
    {
        pos = RemoveOne(succ);
        value = -NegaMax(pos, depth-1);
        if (value > best) best = value;
    }
    return best;
}

int AlphaBeta (pos, depth, alpha, beta)
{
    if (depth == 0) return Evaluate(pos);
    best = -INFINITY;
    succ = Successors(pos);
    while (not Empty(succ) && best < beta)
    {
        pos = RemoveOne(succ);
        if (best > alpha) alpha = best;
        value = -AlphaBeta(pos, depth-1, -beta, -alpha);
        if (value > best) best = value;
    }
    return best;
}
```

# Soteerimise nõks otsingu kiirendamiseks

---

- Sorteerida variandid seisust X enne otsingut ära: alustada tõenäoliselt paremate käikude proovimisega.
  - Suurendab tohutult alpha-beta efekti!
  - Kuidas sorteerida?
- Iteratiivne süvenemine. Teeme:
  - algul täisotsingu sügavuseni 2,
  - siis uue täisotsingu sügavuseni 4,
  - siis uue täisotsingu sügavuseni 6,
  - .... jne ....
  - iga kord kasutame eelmise otsingu tulemust sorteerimiseks!

# Muud standardnõksud

---

- **“Killer moves”**: jätame otsides meelde eriti head käigud:
  - nii masinal kui vastasel
  - proovime kõigepealt varasemast meelde jäetud eriti häid käike
- **“Quiescence search”**: mõnda haru otsitakse sügavamalt:
  - ebastabiilses seisus otsime sügavamalt
  - stabiilses seisus otsime vähem sügavalt
  - otsime lõpuni kõik vahetused ja löögid
- **“Null-Move”**: mis siis, kui vastane saaks kaks käiku järjest?
  - Proovime nii, et vastane saab kaks käiku järjest
  - Kui on meile OK tulemus, siis see on positiivne faktor
  - Kui on meile halb tulemus, jätame meelde “killer move”

# Kuidas mõjub?

---

■ Mida sügavam puu, seda suurem mõju. Näiteks:

■ Käikude puu sügavus viis:

- MiniMax: hindab 10,541,242 seisu
- Alpha-Beta: hindab 1,037,209 seisu
- A-B + “killer moves”: 530,587 seisu.

■ Käikude puu sügavus seitse:

- MiniMax: hindab ca 8,100,000,000 seisu
- Alpha-Beta: hindab 162,662,568 seisu
- A-B + “killer moves”: 46,455,262 seisu.

# Täiendav idee: lõppmängude andmebaasid

## ■ Idee:

- ehitame hiiglasliku lõppmängu-seisude andmebaasi
- igal lõppmängu-seisul on andmebaasis öeldud täpne “headus” (võit, viik, kaotus)
- ehitamiseks: kõigepealt ühe nupuga lõppmängud, siis kahe nupuga lõppmängud, siis kolme nupuga lõppmängud jne.

## ■ Kõigepealt tehti seda 8x8 inglise kabes:

- **Chinook:** Jonathan Schaeffer, Robert Blake, Paul Lu and Martin Bryant:  
University of Alberta

- Kõigi seisude andmebaas, kus on 10 või vähem nuppu

## ■ Sügavad otsingud jõuavad tihti lõppmängude andmebaasini, kus on juba sees täpne seisu hinnang.

## ■ Otsing seega mõlemast suunast!



# Praegune seis mõtlemismängudega

- Lahendatud:
  - Neli nuppu ritta, Qubic,
  - Nine Man's Morris, Go-Moku,
  - Awari
  - Inglise kabe (8x8),
- Tugevalt üle inimmängijate:
  - Renju (viis nuppu ritta),
  - Othello (Reversi), Scrabble, Backgammon
- Maailmameistri tasemel / veidi üle selle:
  - Male,
  - Rahvusvaheline kabe (10x10)
- Tippmängija tasemel:
  - Go (9x9), Hiina male, Bridzh, Pokker
- Amatööri tasemel :
  - Go (19x19)

